Übungszettel 4c - Fehlersuche

Aufgabe 1: Syntaxfehler

Betrachte die folgenden Codeschnipsel. Probiere sie aus und korrigiere die Fehler. *Hinweis: Die Codeschnipsel stehen auf der Vorkursseite zum Download zur Verfügung*

(a) `timeA.py`

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_int)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

(b) `timeB.py`

```
current_time_str = input("What is the current time (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait"

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_str)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

(c) `timeC.py`

```
current_time_str = input("What is the "current time" (in hours 0-23)?")
wait_time_str = input("How many hours do you want to wait")

current_time_int = int(current_time_str)
wait_time_int = int(wait_time_str)

final_time_int = current_time_int + wait_time_int
print(final_time_int)
```

(d) `timeD.py`

```
str_time = input("What time is it now?")
str_wait_time = input("What is the number of hours to wait?")
time = int(str_time)
wai_time = int(str_wait_time)

time_when_alarm_go_off = time + wait_time
print(time_when_alarm_go_off)
```

(e) `timeE.py`

```
n = input("What time is it now (in hours)?")
n = imt(n)
```

```
3 m = input("How many hours do you want to wait?")
4 m = int(m)
5 q = m % 12
6 print("The time is now", q)
```

(f) `timeF.py`

```
1 a = input('wpisz godzine')
2 x = input('wpisz liczbe godzin')
3 int(a)
4 int(x)
5 h = x//24
6 s = x % 24
7 a = a + s
8 print('godzina teraz', a)
```

(g) `timeG.py`

```
1 present_time = input("Enter the present time in hours:")
2 set_alarm = input("Set the hours for alarm:")
3 int (present_time, set_time, alarm_time)
4 alarm_time = present_time + set_alarm
5 print(alarm_time)
```

Solution:

(a) in Zeile 5 steht rechts von der Zuweisung `int(wait_time_int)`. Es muss aber `int(wait_time_str)` heißen, denn `wait_time_int` hat noch gar keinen Wert, kann also auch nicht zugewiesen werden.

(b) Syntaxfehler, angezeigt in Zeile 4, hervorgerufen allerdings durch die nicht geschlossene Klammer am Ende von Zeile 2.

(c) hier gibt's einen Syntaxfehler. In Zeile 1 muss `current time` natürlich Teil des Strings sein. Wenn man es in Anführungszeichen setzen wollte, könnte man dafür lediglich einfache Anführungszeichen nehmen. Die doppelten führen zu einer Beendigung des strings.

(d) in Zeile 4 steht links von der Zuweisung `wai_time` statt `wait_time`

(e) Hier gibt's einen NameError und zwar in Zeile 2. Das Programm ist etwas sinnfrei, da es den Rest der angegebenen Wartezeit mod 5 als aktuelle Zeit ausgibt. Aber das soll uns erstmal nicht stören, es geht nur darum, dass das Programm durchläuft.

(f) Das ist bestimmt etwas ungewohnt, da man hier wenig Anhaltspunkte hat, was das Programm machen soll, es sei denn, man spricht Polnisch. Im Prinzip macht das Programm aber nichts anderes, als die anderen. Nur die Eingabeaufforderung und Ausgabe ist auf Polnisch. Und hier wird versucht das Problem zu beheben, dass das Programm auch einen logischen Fehler enthält. Gibt man nämlich als aktuelle Zeit z.B. 21 ein und möchte 7 Stunden warten, so ist die Ausgabe 28. Was soll das denn bitte für eine Uhrzeit sein??? Hier tritt ein Typfehler auf. Das Problem ist, dass in Zeile 3 und 4 zwar die Eingaben umgewandelt werden sollen, allerdings wird der Rückgabewert in keiner Variablen gespeichert. Die Funktion `int()` verändert den übergebenen Parameter nicht, sondern gibt lediglich den Wert als Integer zurück. Der ursprüngliche String, bleibt davon unberührt. Deshalb "meckert" der Interpreter in Zeile 5,

(g) Das Programm meldet erstmal einen NameError, da `set_time` nicht definiert ist. Ändert man `set_time` in `set_alarm`, tauchen weitere Fehler auf. Ein möglicher fix wäre:

```
1 present_time = input("Enter the present time in hours:")
2 set_alarm = input("Set the hours for alarm:")
3 pt = int(present_time)
```

```
4 sa = int(set_alarm)
5 alarm_time = (pt + sa)%24
6 print(alarm_time)
```

Aufgabe 2: Hier stimmt doch was nicht

Das folgende Programm soll den *Body-Mass-Index (BMI)* für Teilnehmer einer Studie berechnen. Der BMI berechnet sich aus:

$$\frac{\text{Gewicht}[kg]}{\text{Körpergröße}^2[m]}$$

```
1 patients = [[70,1.8], [80, 1.9], [150, 1.7]]
2
3 def calculate_bmi(weight, height):
4     return weight/(height **2)
5
6 for patient in patients:
7     weight, heigth=patients[0]
8     bmi = calculate_bmi(height, weight)
9     print("Patient's BMI is:", bmi)
```

(a) Lade den Quellcode `bmi.py` von der Vorkursseite (http://vorkurs.informatik.uni-frankfurt.de) herunter.

(b) Lass das Programm laufen. Falls der Interpreter Fehler anzeigt, korrigiere diese, sodass du ein lauffähiges Programm erhälst

(c) Die Daten der Teilnehmer sind in der Liste `patients` gespeichert. Kann die Programmausgabe richtig sein?

Solution:

(a)

(b) Startet man das Programm, kommt es in Zeile 8 zu einem NameError. 'height' ist angeblich nicht definiert. Der eigentliche Fehler befindet sich in Zeile 7, dort ist auf der rechten Seite der Zuweisung ein Tippfehler. Statt 'height' steht dort 'heigth'.

Die Leute solle ruhig suchen. Ich habe in der Vorlesung einiges zu NameErrors gesagt. Am besten mit der Suchfunktion des Editors nach 'height' suchen, dann stellt man fest, dass das niemals links von einer Zuweisung steht.

(c) Das Programm gibt 3-mal das gleiche aus. Die Leute sollen nachvollziehen, wie das Programm arbeitet. Vielleicht schreiben sie Kommentare an die Zeilen.... Eigentlich soll für jeden Patienten der BMI berechnet und ausgegeben werden. Da die Patientendaten sehr unterschiedliche Staturen zeigen, ist es verwunderlich, dass alle Patienten den gleichen BMI haben sollten.

Der Fehler ist in Zeile 7, wo immer nur die Daten des ersten Patienten zugewiesen werden (`[patients[0]`) anstatt immer die Daten des aktuellen Patienten `patients[patient]`.

Aufgabe 3: Mäuse

Folgendes Programm soll eine Mäusepopulation simulieren.

```
1 # Eingabe
2 print('Gib die Werte der Ausgangspopulation ein:')
3 jung = input('Anzahl junger Mauese: ')
4 erwachsen = input('Anzahl erwachsener Maeuse: ')
5 alt = ('Anzahl alter Maeuse: '
6 #Verarbeitung
```

```
7  hilf = erwachsen*4 + alt*2
8  alte = erwachsen // 3
9  erwachen = jung // 2
10 jung = hilf
11 #Ausgabe:
12 print('Die berechneten Populationswerte sind:')
13 print('Anzahl junger Maeuse: ', jung)
14 print('Anzahl erwachsenr Maeuse:', erwachsen)
15 print('Anzahl alter Maeuse:', alt)
```

(a) Lade den Quelltext `debugMauese.py` von der Vorkursseite herunter.

(b) Korrigiere das Programm, sodass es ohne Fehlermeldungen läuft.

(c) Teste das Programm. Was soll berechnet werden? Stimmt die Ausgabe?

(d) Ändere das Programm so, dass es auch bei Nutzereingaben die nicht dem vorgesehenen Format entsprechen (ganze Zahlen > 0) nicht abstürzt.

Solution:

Dieser Code enthält sehr viele Fehler. Syntax- und Laufzeitfehler. Wenn man die alle beseitigt hat, stimmt die Ausgabe zur Anzahl der erwachsenen Mäuse noch nicht, (Teil (c)). Hier wird nämlich immer der Eingabewert ausgegeben, obwohl sich die Anzahl der erwachsenen Mäuse aus der Anzahl der jungen Mäuse//2 berechnet. Allerdings ist im Variablennamen auf der linken Seite Zeile 9 ein Tippfehler. Statt `erwachsen` steht dort `erwachen`.
Korrektes Programm:

```
1    # Eingabe
2  print('Gib die Werte der Ausgangspopulation ein:')
3  jung = int(input('Anzahl junger Mauese: '))
4  erwachsen = int(input('Anzahl erwachsener Maeuse: '))
5  alt = int(input('Anzahl alter Maeuse: '))
6  #Verarbeitung
7  hilf = erwachsen*4 + alt*2
8  alte = erwachsen // 3
9  erwachsen = jung // 2
10 jung = hilf
11 #Ausgabe:
12 print('Die berechneten Populationswerte sind:')
13 print('Anzahl junger Maeuse: ', jung)
14 print('Anzahl erwachsener Maeuse:', erwachsen)
15 print('Anzahl alter Maeuse:', alt)
```

(d) können die Leute machen wie sie wollen. Mit `while True:` oder `try:` oder was ihnen auch einfällt.

Aufgabe 4: Programmieraufgabe: Euklidischer Algorithmus

Der euklidische Algorithmus ist ein Verfahren den größten gemeinsamen Teiler (ggT) zweier Zahlen zu ermitteln.

Man teilt die größere durch die kleinere Zahl. Geht die Division auf, ist der Divisor der ggT. Geht die Division nicht auf, bleibt ein Rest. Dieser Rest ist der neue Divisor. Der alte Divisor wird zum Dividenden. Nach endlich vielen Schritten erhält man den ggT.

(a) Schreibe eine Funktion, die den ggT zweier übergebener Zahlen berechnet.

(b) Schreibe ein Programm, das den Benutzer natürliche Zahlen eingeben lässt und diese in einer Liste speichert.

- Überlege dir, wie der Benutzer dem Programm mitteilen soll, dass er/sie keine weiteren Werte mehr eingeben möchte.
- Ist die Eingabe vollständig, soll das Programm den ggT aller benachbarten Zahlen in der Liste berechnen und diese ausgeben.

- nutze dafür die in (a) programmierte Funktion, indem du sie als Modul importierst.

Beispiel Wenn die Zahlen 27, 9, 33, 1987 und 3195 eingegeben werden, soll das Programm folgende Ausgabe haben:
ggT von 27 und 9 : 9
ggT von 9 und 33 : 3
ggT von 33 und 1987 : 1
ggT von 1987 und 3195 : 1

Alternativ kannst du auch das Programm `debugGgTProg.py` und die Funktion `debugEuklid.py` von der Webseite herunterladen und debuggen.

Solution:

(a)
```python
def ggT(b,c):
    y,x = b,c

    while True:
        remainder = x%y
        if not remainder:
            break
        x = y
        y = remainder
    return y
```

(b)
```python
import euklid

done = False
numbers = [] #leere Liste für Zahlen anlegen
#Eingabe
while not done:
    while True:
        a = input("bitte ganze Zahl > 0 eingeben: ")
        if a =="q":
            done = True # Eingabe beendet
            break
        elif a.isdigit():
            numbers.append(int(a)) # in Zahl umwandeln und in Liste
                schreiben
# ggTs berechnen
ggTs = [] # leere Liste für ggTs
for i in range(len(numbers)-1):
    ggT = euklid.ggT(numbers[i],numbers[i+1])#ggT benachbarter Zahlen
        berechnen
    ggTs.append(ggT)

#Ausgabe
for i in range(len(ggTs)):
    print("ggT von",numbers[i],"und", numbers[i+1],":",ggTs[i])
```

Wenn die Leute den Code debuggen wollen, sieht das korrekte Programm natürlich so aus wie oben. Achtung, sowohl debugEuklid, als auch das Programm enthalten Fehler. Importieren muss man dann natürlich das Modul debugEuklid und nicht euklid.

Aufgabe 5: Programmieraufgaben

Falls Sie weitere Programmieraufgaben brauchen zum Üben:

(a) Schreibe ein Programm, das alle Zahlen zwischen 2000 und 3200 (einschließlich) die ein Vielfaches von 7, aber nicht von 5 sind. Die Zahlen sollen in einer Zeile, durch Kommata getrennt ausgegeben werden.

(b) Schreibe ein Programm, das Werte nach folgender Formel berechnet:

$$Q = \sqrt{\frac{2 \cdot C \cdot D}{H}}$$

Dabei sind H und C Konstanten, mit den Werten $C = 50$ und $H = 30$. D ist eine Variable. Der Nutzer soll die Möglichkeit haben gleich mehrere Werte mit Kommata getrennt über die Konsole einzugeben. Die Ausgabe soll ebenfalls als durch Kommata getrennte Werte erfolgen.
Beispiel:
Für die Eingabe `100,150,180`, soll das Programm folgende Ausgabe liefern: `18,22,24`
Hinweis: Die Methode `split()` für `strings` könnte hilfreich sein bei der Verarbeitung der Eingabe.

Solution:

(a)
```
l=[]
for i in range(2000, 3201):
    if (i%7==0) and (i%5!=0):
        l.append(str(i))

ausgabe = str(l[0])
for i in range(1,len(l)):
    ausgabe = ausgabe + "," + str(l[i])
print(ausgabe)
```

(b)
```
import math
c=50
h=30
value = []
items=[x for x in input().split(',')]
for d in items:
    value.append(str(int(round(math.sqrt(2*c*float(d)/h)))))

print (','.join(value))
```

Viel Erfolg!